

SSL: Secure Socket Layer

Crittografia

Luciano Margara

Unibo

2022

Introduzione

I servizi web quali Internet banking, E-commerce e molti altri sono ormai diffusissimi e richiedono in genere lo scambio via rete di importanti dati riservati, per esempio il numero della propria carta di credito. Poiché l'intercettazione di questi dati potrebbe esporre gli utenti a gravi danni, il sistema che offre il servizio via Internet deve garantire opportune contromisure per assicurare che la comunicazione avvenga in modo altamente confidenziale

Introduzione

D'altro canto molti utenti ignorano i meccanismi che sono alla base di questa comunicazione, assumendo un atteggiamento tra fatalista e fiducioso nella competenza e correttezza di chi fornisce il servizio. Cerchiamo quindi di fare un po' di chiarezza su come gli strumenti crittografici fin qui presentati possano essere combinati per definire protocolli sicuri, tra cui prendiamo a esempio il diffusissimo Secure Socket Layer (brevemente SSL) che garantisce la confidenzialità e l'affidabilità delle comunicazioni su Internet

Scenario

Consideriamo un utente U che desidera accedere tramite Internet a un servizio offerto da un sistema S

Confidenzialità

Il protocollo SSL garantisce la confidenzialità poiché la trasmissione è cifrata mediante un sistema ibrido, in cui un cifrario asimmetrico è utilizzato per costruire e scambiare le chiavi di sessione, e un cifrario simmetrico utilizza queste chiavi per criptare i dati nelle comunicazioni successive.

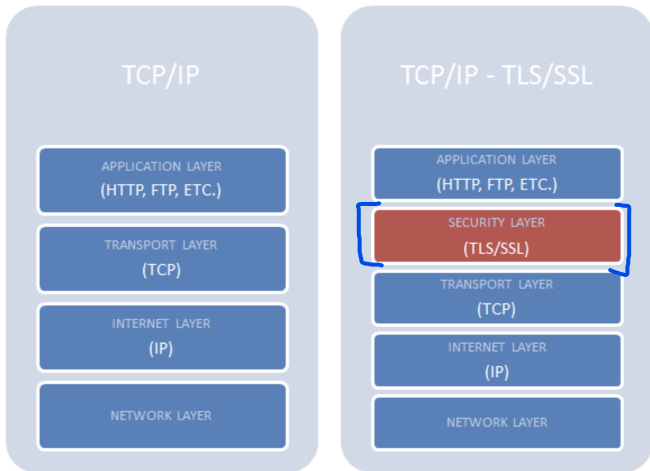
Autenticazione

Il protocollo garantisce inoltre la autenticazione dei messaggi accertando l'identità dei due partner, attraverso un cifrario asimmetrico o facendo uso di certificati digitali, e inserendo nei messaggi stessi un apposito MAC che utilizza una funzione hash one-way crittograficamente sicura

Stack di protocolli

SSL si innesta tra un protocollo di trasporto affidabile, per esempio TCP/IP, e un protocollo di applicazione, per esempio HTTP. SSL è completamente indipendente dal protocollo di applicazione sovrastante che può utilizzarlo in maniera trasparente all'utente

SSL nello stack dei protocolli di rete



SSL Record

SSL è organizzato su due livelli. Al livello più basso risiede il protocollo SSL Record che è connesso direttamente al protocollo di trasporto e ha l'obiettivo di incapsulare i dati spediti dai protocolli dei livelli superiori assicurando la confidenzialità e l'integrità della comunicazione.

SSL Handshake

Al livello superiore risiede il protocollo SSL Handshake che è interfacciato all'applicazione sovrastante e permette all'utente e al sistema di autenticarsi, di negoziare gli algoritmi di cifratura e firma, e di stabilire le chiavi per i singoli algoritmi crittografici e per il MAC

SSL Handshake + Record

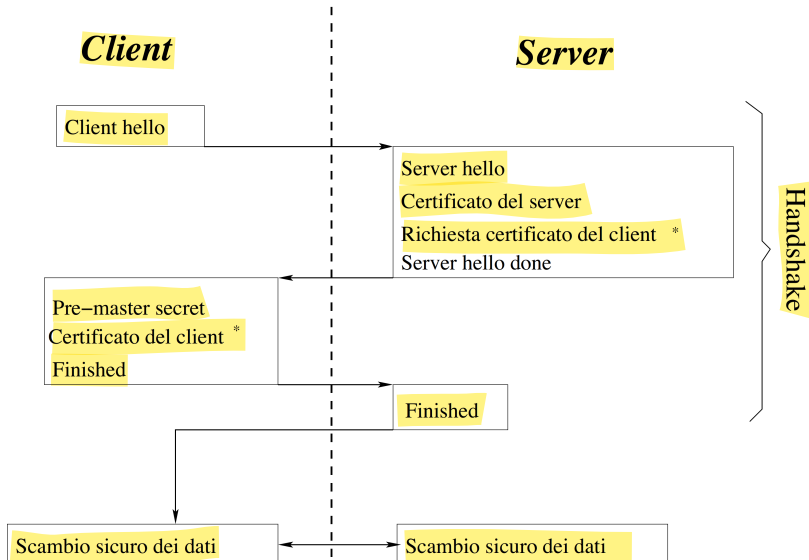
In sostanza SSL Handshake crea un canale sicuro, affidabile e autenticato tra utente e sistema, entro il quale SSL Record fa viaggiare i messaggi incapsulandoli in blocchi opportunamente cifrati e autenticati.

SSL Handshake

Una sessione di comunicazione SSL è innescata da uno scambio di messaggi preliminari che prende il nome di **handshake** (stretta di mano) come il protocollo che lo governa.

Attraverso questi messaggi il sistema S (server) e l'utente U (client) si identificano a vicenda e cooperano alla costruzione delle chiavi segrete da impiegare nelle comunicazioni simmetriche successive: costruzione che procede da un'unica informazione segreta detta **master secret** su cui l'utente e il sistema si accordano in modo incrementale

Handshake



1- Utente: client hello 1/2

L'utente U manda al sistema S un messaggio, detto client hello, con cui richiede la creazione di una connessione SSL, specifica le prestazioni di sicurezza che desidera siano garantite durante la connessione, e invia una sequenza di byte casuali. In dettaglio il messaggio di client hello specifica la versione del protocollo SSL eseguito da U , un elenco di algoritmi di compressione, e una lista di cipher suite, cioè di cifrari e meccanismi di autenticazione che U può supportare ordinati secondo le sue preferenze

1- Utente: client hello 2/2

Per esempio la cipher suite indicata con la sigla **SSL RSA WITH 2TDEA EDE CBC SHA1** prevede RSA per lo scambio delle chiavi di sessione. 2TDEA EDE CBC per la cifratura simmetrica, ove EDE indica la sequenza encryption-decryption-encryption per il Des Triplo con due chiavi 2TDEA e CBC indica l'impiego di un cifrario a composizione di blocchi; e prevede SHA1 come funzione hash one-way per il calcolo dei MAC

2- Sistema: server hello

Il sistema S riceve il messaggio di client hello e seleziona da questo un cipher suite e un algoritmo di compressione che anch'esso è in grado di supportare. Dopodiché invia all'utente un messaggio, detto server hello, che specifica la sua scelta e contiene una nuova sequenza di byte casuali. Se U non riceve il messaggio di server hello, interrompe la comunicazione.

3- Sistema: invio del certificato

Solitamente il sistema si autentica con l'utente inviandogli il proprio certificato digitale. Esistono più Certification Authority (CA) possibilmente organizzate in una struttura ad albero, e non è detto che U e S si affidino alla stessa CA per l'emissione dei loro certificati. In questo caso il sistema invia all'utente una sequenza di certificati ordinati in accordo alla CA che li ha firmati, da quella che ha certificato S a quella che è alla radice dell'albero delle CA. Nel caso in cui i servizi offerti da S debbano essere protetti negli accessi, il sistema può richiedere all'utente di autenticarsi inviando il suo certificato digitale

4- Sistema: server hello done

Il sistema invia il messaggio server hello done con il quale sancisce la fine degli accordi sul cipher suite e sui parametri crittografici ad essa associati

5- Utente: autenticazione del sistema

Per accertare l'autenticità del certificato ricevuto dal sistema, U controlla che la data corrente sia inclusa nel periodo di validità del certificato, quindi il certificato non è scaduto che la CA che ha firmato il certificato sia tra quelle fidate e che la firma da essa apposta sia autentica. Nel caso che S invii una lista di certificati, U esegue la relativa catena di verifiche.

6- Utente: invio del pre-master secret e costruzione del master secret 1/2

L'utente U costruisce un pre-master secret costituito da una nuova sequenza di byte casuali, lo cifra con il cifrario a chiave pubblica del cipher suite su cui si è accordato con S e spedisce il relativo crittogramma a S . Con riferimento alla cipher suite presa in esempio al passo 1, U cifra il pre-master secret con l'algoritmo RSA utilizzando la chiave pubblica presente nel certificato di S

6- Utente: invio del pre-master secret e costruzione del master secret 2/2

Il pre-master secret viene inoltre combinato da U con alcune stringhe note e con i byte casuali presenti sia nel messaggio di client hello che in quello di server hello. A tutte queste sequenze l'utente applica le funzioni hash one-way SHA1 e MD5 secondo una combinazione opportuna. Il risultato costituisce il master secret

7- Sistema: ricezione del pre-master secret e costruzione del master secret

Il sistema decifra il crittogramma contenente il pre-master secret ricevuto dall'utente e calcola il master secret mediante le stesse operazioni eseguite dall'utente nel passo 6, in quanto dispone delle stesse informazioni

8- Utente: invio del certificato (opzionale)

Se all'utente viene richiesto un certificato (passo 3) ed egli non lo possiede, il sistema interrompe l'esecuzione del protocollo. Altrimenti l'utente invia il proprio certificato con allegate una serie di informazioni firmate con la sua chiave privata, che contengono tra l'altro il master secret che egli stesso ha generato e tutti i messaggi scambiati fino a questo momento (SSL-history). S controlla il certificato di U con operazioni simmetriche a quelle eseguite da U nel passo 5, e verifica l'autenticità e la correttezza della SSL-history. In presenza di anomalie la comunicazione con U viene interrotta.

9- Utente e Sistema: messaggio finished 1/2

Questo messaggio è il primo ad essere protetto mediante il master secret e il cipher suite su cui i due partner si sono accordati. Il messaggio viene prima costruito dall'utente e spedito al sistema, poi costruito dal sistema e spedito all'utente: nei due invii la struttura del messaggio è la stessa ma cambiano le informazioni in esso contenute. La sua costruzione consta di due passi. All'inizio si concatenano il master secret, tutti i messaggi di handshake scambiati fino a quel momento e l'identità del mittente (U oppure S)

9- Utente e Sistema: messaggio finished 2/2

La stringa così ottenuta viene trasformata applicando le funzioni SHA1 e MD5, dando origine a una coppia di valori che costituisce il messaggio finished. Questo messaggio è diverso nelle due comunicazioni poiché S aggiunge ai messaggi di handshake anche il messaggio finished ricevuto dall'utente. Il destinatario della coppia (U oppure S) non può invertire la computazione precedente in quanto generata da funzioni one-way, ma ricostruisce l'ingresso delle due funzioni SHA1 e MD5, ricalcola queste funzioni e controlla che la coppia così generata coincida con quella ricevuta, come dimostrazione che la comunicazione è avvenuta correttamente

Master secret

La "sequenza di inizializzazione" è il vettore di inizializzazione (IV), che introduce pseudo-casualità nel processo di cifratura, garantendo che messaggi identici producano output cifrati diversi.

Il master secret viene utilizzato da U e da S per costruire una propria tripla contenente¹ la chiave segreta da adottare nel cifrario simmetrico (per esempio 3DES EDE)², la chiave da adottare in una funzione hash one-way per la costruzione del MAC (per esempio SHA1), e³ la sequenza di inizializzazione per cifrare in modo aperiodico messaggi molto lunghi (usata per esempio nel CBC). Le triple dell'utente e del sistema sono diverse tra loro ma note a entrambi i partner: ciascuno usa la propria, il che aumenta la sicurezza delle comunicazioni

Canale sicuro

Il canale sicuro approntato dal protocollo SSL Handshake viene realizzato dal protocollo SSL Record. I dati sono frammentati in blocchi di lunghezza opportuna. Ciascun blocco viene numerato, compresso, autenticato attraverso l'aggiunta di un MAC, cifrato mediante il cifrario simmetrico su cui l'utente e il sistema si sono accordati, e finalmente trasmesso da SSL Record utilizzando il protocollo di trasporto sottostante. Il destinatario della comunicazione esegue un procedimento inverso sui blocchi ricevuti: decifra e verifica la loro integrità attraverso il MAC, decomprime e riassume i blocchi in chiaro, infine li consegna all'applicazione sovrastante.

Sistema senza certificato

Se il sistema S non dispone di un certificato, il protocollo di SSL Handshake può ancora essere eseguito se i due partner si servono di un algoritmo asimmetrico per lo scambio segreto del pre-master secret, quale ad esempio quello del protocollo DH che dovrà essere specificato nel cypher suite. In questo caso però, come abbiamo già osservato, non si ha protezione dall'intrusione di un crittoanalista attivo che esegua un attacco tipo man-in-the-middle

In assenza di un certificato, il protocollo può comunque procedere utilizzando un algoritmo asimmetrico, come il Diffie-Hellman (DH), che è incluso nel cypher suite scelto. Con DH, client e server sono in grado di calcolare un segreto comune (il pre-master secret) attraverso lo scambio di valori pubblici, senza bisogno di una chiave privata del server certificata e resa nota:

-> Entrambe le parti generano un proprio valore segreto e un corrispondente valore pubblico. Scambiandosi i valori pubblici, ognuno calcola il pre-master secret combinando il proprio segreto con il valore pubblico ricevuto. Il risultato è un segreto comune, che verrà usato per derivare ulteriori chiavi di sessione.

Client hello e server hello

Nei passi di hello i due partner creano e si inviano due sequenze casuali per la costruzione del master secret, che risulta così diverso in ogni sessione di SSL. Questo impedisce a un crittoanalista di riutilizzare i messaggi di handshake catturati sul canale in una sessione precedente, per sostituirsi a S in una successiva comunicazione con U (attacco di reply)

MAC dei blocchi di dati 1/2

Il protocollo SSL Record numera in modo incrementale ogni blocco di dati e autentica il blocco attraverso un MAC. Per prevenire la modifica del blocco da parte di un crittoanalista attivo, il MAC viene calcolato come immagine hash one-way di una stringa costruita concatenando il contenuto del blocco, il numero del blocco nella sequenza, la chiave del MAC, e alcune stringhe note e fissate apriori

MAC dei blocchi di dati 2/2

MAC per integrità ed autenticazione

Se un blocco di dati venisse intercettato e riproposto (un attacco di replay), il numero del blocco incluso nel calcolo del MAC non corrisponderebbe alla sequenza attesa. Ad esempio, se un blocco "X" viene inviato come blocco 3, un attaccante non può semplicemente replicare lo stesso blocco "X" in un'altra parte della sessione: il destinatario si accorgerà del mismatch rispetto all'ordine sequenziale previsto.



La specificazione del numero del blocco consente di prevenire l'impiego fraudolento e iterato del blocco stesso nell'ambito della stessa sessione (attacco di replay), perché si dovrebbe cambiare quel numero e il MAC corrispondente. Poiché i MAC sono cifrati insieme al messaggio, un crittoanalista non può alterarli senza aver forzato prima la chiave simmetrica di cifratura: quindi un attacco volto a modificare la comunicazione tra i due partner è difficile almeno quanto quello volto alla sua decrittazione.

Autenticazione del Sistema

Il canale definito dal protocollo SSL Handshake è immune da attacchi del tipo man in-the-middle poiché il sistema viene autenticato con un certificato digitale. L'utente ha così la possibilità di comunicare il premaster secret al sistema in modo sicuro attraverso la chiave pubblica presente nel certificato di S . Solo S può decifrare quel crittogramma e quindi costruire il corretto master secret, su cui si fonda la costruzione di tutte le chiavi segrete adottate nelle comunicazioni successive. Quindi solo il sistema S , ossia quello a cui si riferisce il certificato, potrà entrare nella comunicazione con l'utente U

Autenticazione dell'Utente

L'utente può essere autenticato. Il certificato dell'utente (se richiesto) e la sua firma apposta sui messaggi scambiati nel protocollo (SSL-history) consentono al sistema di verificare che l'utente è effettivamente quello che dichiara di essere e che i messaggi sono stati effettivamente spediti da esso. Se ciò non si verifica, il sistema deduce che il protocollo è stato alterato casualmente o maliziosamente e interrompe la comunicazione

Autenticazione dell'Utente

Sottolineiamo come l'opzionalità dell'autenticazione dell'utente sia uno dei fattori che hanno favorito la diffusione di SSL e di altri protocolli simili nelle transazioni commerciali via Internet. Infatti mentre i sistemi sono solitamente creati da società che possono facilmente ottenere certificati da parte di CA fidate, per gli utenti, spesso singoli privati, la necessità di certificazione può costituire un serio ostacolo pratico ed economico. Se l'autenticazione dell'utente è comunque necessaria per garantire la protezione di alcuni servizi, si preferisce procedere creando il canale sicuro attraverso SSL e poi autenticando l'utente con metodi tradizionali, per esempio mediante login e password

Il messaggio finished

Questo messaggio viene costruito in funzione del master secret e contiene tutte le informazioni che i due partner si sono scambiati nel corso dell'handshake. Lo scopo è quello di consentire ai due partner di effettuare un ulteriore controllo sulle comunicazioni precedenti per garantire che queste siano avvenute correttamente, che essi dispongano dello stesso master secret, e che la loro comunicazione non sia stata oggetto di un attacco attivo

Generazione delle sequenze casuali

Le tre sequenze casuali generate dall'utente e dal sistema e comunicate nei messaggi di client hello, server hello e pre-master secret entrano crucialmente in gioco nella creazione del master secret e quindi nella generazione delle chiavi di sessione. In particolare la sequenza corrispondente al pre-master secret viene generata dall'utente e comunicata per via cifrata al sistema. La non prevedibilità di questa sequenza è uno dei fulcri su cui si fonda l'intera sicurezza del canale SSL: una sua cattiva generazione renderebbe il protocollo molto debole, ribadendo l'importanza che rivestono i generatori pseudo-casuali nei processi crittografici.

Sicurezza di SSL

Il protocollo SSL è sicuro almeno quanto il più debole cipher suite da esso supportato. Poiché, dopo i tentativi di alcuni governi di limitare l'impiego della crittografia su Web per motivi di sicurezza nazionale, dal Gennaio 2000 le norme internazionali non pongono alcuna limitazione sui cifrari impiegabili se non in alcuni paesi che si trovano in condizioni politiche particolari, è consigliabile disabilitare i propri sistemi dall'impiego di cifrari ormai notoriamente insicuri o di chiavi troppo corte.

SSL \rightarrow TLS

SSL (Secure Sockets Layer):

Sviluppato da: Netscape nel 1994 per proteggere le comunicazioni HTTP

Versioni:

SSL 1.0: mai pubblicato per gravi problemi di sicurezza

SSL 2.0 (1995): prima versione pubblica. Inadeguata: mancava autenticazione dei messaggi

SSL 3.0 (1996): completamente ridisegnato. Più sicuro, ma ancora vulnerabile. È stato il punto di partenza per TLS

SSL è oggi obsoleto e non più supportato nei browser e nei server moderni.

SSL \Rightarrow TLS

TLS (Transport Layer Security)

Standardizzato da: IETF come aggiornamento sicuro di SSL

Versioni:

TLS 1.0 (1999): basato su SSL 3.0, ma con cifrature migliorate

TLS 1.1 (2006): aggiunta protezione contro il CBC padding oracle attack

TLS 1.2 (2008): supporto per algoritmi moderni come AES-GCM e SHA-256. Ancora molto diffuso oggi

TLS 1.3 (2018): riduce handshake, rimuove vecchi algoritmi (RSA key exchange, MD5, SHA-1), migliora la velocità e la sicurezza. È la versione più sicura e veloce ad oggi

TLS 1.2 e 1.3 sono gli unici realmente sicuri e supportati nei sistemi moderni.

IETF

È l'organismo internazionale responsabile dello sviluppo e della standardizzazione dei protocolli Internet, tra i quali: TLS, HTTP, TCP/IP, DNS, SMTP (email)

Sviluppa e pubblica specifiche tecniche chiamate RFC (Request for Comments), collabora in modo aperto e trasparente, non è un ente governativo né a scopo di lucro

Esempio: TLS 1.3 è definito nell'RFC 8446, pubblicato dall'IETF

IETF

Organizzazione:

Lavoro suddiviso in Working Groups (es. "tls", "httpbis")

Supervisione generale da parte dell'IESG (Internet Engineering Steering Group)

Filosofia:

Aperta a chiunque voglia partecipare

Motto: "Rough consensus and running code" (consenso informale e codice funzionante)

CBC

CBC è una modalità di cifratura dove i dati sono divisi in blocchi, ogni blocco cifrato dipende dal blocco precedente, alla fine, i dati devono essere "riempiti" (padding) se non sono multipli della lunghezza del blocco

CBC Padding Oracle Attack

In modalità CBC, il messaggio in chiaro viene diviso in blocchi di dimensione fissa. Ogni blocco viene combinato (tipicamente tramite un'operazione XOR) col blocco cifrato precedente prima di essere cifrato, con l'eccezione del primo blocco, che viene combinato con un vettore di inizializzazione (IV).

È un attacco che sfrutta errori di risposta del server per decifrare dati cifrati, anche senza conoscere la chiave

Il server usa una cifratura CBC, che aggiunge "riempimento" (padding) ai dati.

L'attaccante invia messaggi modificati.

Se il server dice "padding errato", l'attaccante capisce informazioni sui dati cifrati.

Ripetendo il trucco, può decifrare tutto il messaggio.

L'attaccante intercetta un messaggio cifrato e lo modifica su uno o più blocchi. In particolare, modifica il blocco precedente a quello che si vuole decifrare. Dopo aver inviato il messaggio alterato, l'attaccante osserva la risposta del server. Se il server riporta un errore specifico (ad esempio, "padding errato"), questo indica che il blocco modificato non ha generato un padding valido. Con una serie di tentativi e errori, variando in modo sistematico i byte del blocco modificato, l'attaccante riesce a determinare quali modifiche portano a un padding valido. Questo processo permette di dedurre il valore intermedio che, combinato con il blocco successivo e, infine, con un'operazione XOR, consente di recuperare il blocco in chiaro. Attraverso ripetuti tentativi su ogni blocco del messaggio, l'attaccante può decifrare l'intero messaggio senza avere la chiave segreta, sfruttando esclusivamente le risposte del sistema.

AES-GCM

- AES: Advanced Encryption Standard è un algoritmo di cifratura simmetrica, sicuro e veloce
- GCM sta per Galois/Counter Mode: modalità che cifra e autentica i dati insieme (AEAD)
- AEAD (Authenticated Encryption with Associated Data) è una modalità di cifratura moderna che protegge i dati in due modi contemporaneamente: rende il contenuto illeggibile a chi non ha la chiave e permette di verificare che i dati non siano stati modificati.

RSA key exchange e forward secrecy

Il server invia al client il certificato digitale, che contiene la chiave pubblica RSA, il client genera una chiave di sessione (segreta), il client cifra la chiave con la chiave pubblica RSA del server, il server decifra con la sua chiave privata e ottiene la chiave di sessione.

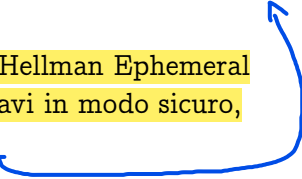
La chiave di sessione è stata trasmessa (anche se cifrata) e RSA non cambia chiavi ogni volta, ma usa sempre la stessa chiave pubblica/privata.

Usando ECDHE, dove: La chiave non viene inviata, ma viene calcolata insieme, e mai trasmessa e soprattutto è diversa per ogni sessione. Se ne viene violata una, non si violano tutte le altre.

ECDHE

Grazie all'uso di chiavi "ephemeral", anche se in futuro una chiave privata venisse compromessa, le sessioni passate rimangono sicure perché ogni sessione ha usato una chiave diversa. Questo è il principio della forward secrecy, che assicura che la compromissione di una singola chiave non comprometta l'intera cronologia delle comunicazioni.

ECDHE sta per Elliptic Curve Diffie-Hellman Ephemeral
È un metodo usato per scambiare chiavi in modo sicuro,
garantendo anche la Forward Secrecy



Diffie-Hellman: metodo per generare una chiave segreta
condivisa tra due parti senza trasmetterla

Elliptic Curve: versione moderna, più efficiente e sicura

Ephemeral: ogni sessione usa nuove chiavi temporanee e quindi
ogni connessione è unica

TLS: differenze principali rispetto a SSL


Negoziiazione iniziale

(TLS) Il client invia ClientHello con:

Versione TLS supportata (solo TLS \geq 1.2)

Cipher suites moderne (es. AES-GCM, ChaCha20)

Parametri di key exchange (KeyShare, es. ECDHE)

Session ticket (per 0-RTT, opzionale) 

Si inviano, insieme a ClientHello, altre informazioni per evitare di mandare altri messaggi dopo, velocizzando la Negoziiazione Iniziale, legate alle sessioni precedenti

(SSL) In SSL, la versione è SSLv2/3, e i key exchange possono usare RSA diretto (non forward-secret) e le suite includono algoritmi obsoleti.

ChaCha20

ChaCha20 è un algoritmo di cifratura simmetrica, veloce e sicuro, spesso usato come alternativa ad AES, soprattutto su dispositivi mobili o meno potenti

È uno stream cipher: cifra i dati bit per bit o byte per byte

Creato da Daniel J. Bernstein

Spesso usato insieme a Poly1305, per autenticare i dati insieme formano ChaCha20-Poly1305, un algoritmo AEAD (Authenticated Encryption with Associated Data)

Poly1305

Poly1305 è un algoritmo di autenticazione dei messaggi, usato per garantire che i dati non siano stati modificati durante la trasmissione.

Poly1305 non cifra i dati, ma crea un "tag" di verifica (simile a una firma) che assicura che il messaggio sia integro, permette al destinatario di verificare l'autenticità e impedisce modifiche o manomissioni invisibili

0-RTT

0-RTT (zero round-trip time) è una funzionalità di TLS 1.3 che permette al client di inviare dati cifrati fin dal primo messaggio, senza aspettare la risposta del server

Il client ha già avuto una connessione TLS con quel server in passato,

salva un session ticket fornito dal server,

alla riconnessione invia ClientHello + subito i dati 0-RTT (es. richiesta HTTP),

il server risponde normalmente (ServerHello, ecc.) e può accettare o rifiutare i dati 0-RTT

TLS: differenze principali rispetto a SSL

Risposta del server

(TLS) Il server risponde con:

Scelta di versione e suite

Parametri per la key exchange (ECDHE)

EncryptedExtensions: informazioni come ALPN o SNI

Certificate: certificato del server (X.509)

CertificateVerify: firma del certificato

Finished: MAC autenticato sull'handshake

in TLS 1.3 tutti i messaggi dopo ServerHello sono cifrati mentre SSL usava messaggi in chiaro fino al termine dell'handshake.

ALPN o SNI

consente al client e server di concordare, durante il TLS handshake, quale protocollo applicativo utilizzare senza dover effettuare un ulteriore round-trip

ALPN (Application-Layer Protocol Negotiation) permette al client e server di scegliere quale protocollo applicativo usare dopo TLS (es. HTTP/1.1 o HTTP/2), evita round-trip extra per negoziare il protocollo, rendendo più veloce la connessione. Il client dice: "so parlare HTTP/2 e HTTP/1.1" e il server risponde: "usiamo HTTP/2". La negoziazione avviene durante il TLS handshake

per consentire al client di indicare al server quale nome di dominio intende raggiungere, prima che venga completata la fase di stabilimento del canale TLS.

SNI (Server Name Indication) permette al client di dire a quale dominio vuole collegarsi, prima che la connessione TLS venga stabilita, su un server che ospita più siti web (hosting condiviso), il server deve sapere quale certificato TLS usare, il client include il nome del dominio nel messaggio ClientHello e il server seleziona il certificato corretto per quel dominio

SNI rende possibile la gestione di certificati per più domini sullo stesso server, facilitando il moderno ambiente di hosting condiviso e garantendo che la corretta configurazione TLS venga scelta fin dall'inizio del handshake.

TLS: differenze principali rispetto a SSL

Finalizzazione

Il client:

Verifica il certificato

Invia Finished

Da qui, entrambe le parti possono scambiarsi Application Data
cifrato

TLS: differenze principali rispetto a SSL

Cifratura e sicurezza

Solo algoritmi AEAD (Authenticated Encryption with Associated Data)

Nessuna supporto per RSA key exchange, CBC mode, MD5, SHA-1, Compressione (vulnerabile a CRIME)

TLS 1.3 è più sicuro per design: meno opzioni = meno possibilità di errore

SSL era estendibile ma fragile, e ha sofferto numerose vulnerabilità (es. POODLE, BEAST)

CRIME (Compression Ratio Info-leak Made Easy)

Un attacco che sfrutta la compressione TLS per rubare cookie o token di sessione,
il browser comprime i dati prima di cifrarli e l'attaccante osserva quanto "accorcia" il messaggio,
con ripetuti tentativi, indovina i dati segreti (es. cookie),
colpiva TLS con compressione abilitata,
soluzione: disattivare la compressione TLS (oggi è disabilitata di default)

POODLE (Padding Oracle On Downgraded Legacy Encryption)

Un attacco contro SSL 3.0, sfrutta errori di padding nel blocco cifrato (CBC mode),
forza il client a usare SSL 3.0 (downgrade),
modifica i blocchi cifrati,
osserva la risposta del server e riesce a rubare dati come i cookie,
colpiva SSL 3.0 e TLS se mal configurato,
soluzione: disattivare SSL 3.0 e usare modalità AEAD (es. AES-GCM)

BEAST (Browser Exploit Against SSL/TLS)

Attacco contro TLS 1.0 e CBC mode, sfrutta l'ordine di cifratura dei blocchi per decifrare cookie in sessione, inietta codice maligno nel browser (JavaScript), cattura i blocchi cifrati, ricostruisce byte dopo byte il cookie HTTPS, colpiva TLS 1.0 con CBC e browser vulnerabili, soluzione: aggiornare a TLS 1.2+ e usare AES-GCM o ChaCha20-Poly1305

TLS: differenze principali rispetto a SSL

Performance

Handshake ridotto a 1-RTT

Supporto opzionale per 0-RTT (dati inviati subito al primo pacchetto, utile in sessioni ripetute)

Meno round-trip e quindi più velocità

SSL richiedeva handshake a più fasi, più lento e meno efficiente

Vantaggi di TLS rispetto a SSL

- **Maggiore sicurezza**: algoritmi moderni, protezione contro attacchi noti (es. POODLE, BEAST)
- **Più veloce**: handshake rapido (1-RTT) e supporto per 0-RTT
- **Forward Secrecy**: protezione anche se la chiave privata viene compromessa
- **Meno complessità, meno errori**: rimozione di cifrature deboli e opzioni obsolete
- **Cifratura precoce**: i messaggi sono protetti fin dall'inizio della connessione
- **Ripresa efficiente**: connessioni successive più rapide e sicure
- **Standard moderno e supportato**: compatibile con browser, server e dispositivi aggiornati